DOCUMENTATION

Imports and Model Setup:

- Various libraries are imported, including os, numpy, PIL for image processing, cv2 for OpenCV, and Flask for the web app.
- VGG19, a pre-trained CNN model, is used with custom dense layers on top. The model excludes the top layer and uses input images of size (240, 240, 3).
- A few fully connected layers are added for classification, ending with a softmax layer for binary output ("No Brain Tumor" or "Yes Brain Tumor").
- The model weights are loaded from the file 'vgg_unfrozen.h5'.

Functions:

- get_className(classNo): Maps the predicted class number (0 or 1) to a corresponding label ("No Brain Tumor" or "Yes Brain Tumor").
- getResult(img): Reads an image file, resizes it to the input shape (240, 240), and uses the VGG19 model to predict whether it shows a brain tumor. The predicted class is returned.

Flask Routes:

- '/': Serves the index.html page, which likely contains the interface for users to upload images.
- /predict: Handles file uploads via a POST request. The uploaded file is saved, and the getResult() function is used to predict if the image contains a brain tumor. The result is returned as a string.

Running the App:

- The app runs locally on http://127.0.0.1:5000/ in debug mode.

The working of ipynb file:

train_datagen:

- **This generator is used to augment the training data to prevent overfitting and improve generalization. It performs several transformations.**

test_data_gen and valid_data_gen:

- **These generators are for test and validation data. They only apply rescaling to normalize pixel values (1.0/255), without any augmentation, to ensure the original test/validation data is preserved for accurate model evaluation.**